

Architectural Power Management for High Leakage Technologies

Manish Kulkarni, Khushboo Sheth and Vishwani D. Agrawal

Auburn University, Department of Electrical and Computer Engineering, Auburn, AL 36849, USA

mmk0002@auburn.edu, shethkh@auburn.edu, vagrawal@eng.auburn.edu

Abstract

We propose a power-performance trade off methodology for microprocessors. An instruction named slowdown for low power (SLOP) is introduced. Functionally, it resembles the conventional NOP but requires power-specific hardware implementation. Depending upon the power reduction requirement, adequate number of SLOP's are automatically inserted in the instruction stream by the power management hardware. While processing a SLOP, additional power control signals are generated for various units; the ALU is powered down, caches are put in drowsy mode, and register file and pipeline registers may be fully or partially clock-gated. Simulation of a five-stage pipelined 32-bit MIPS processor shows that the SLOP method, termed instruction slowdown (ISD), becomes more effective than a conventional clock slowdown (CSD) when leakage is high. For 32nm CMOS technology, ISD can save more than 70% power compared to about 40% by CSD. The paper shows that power reduction through a judicious choice of slowdown factor and the method adopted, clock slowdown for low leakage and instruction slowdown for high leakage, can enhance the battery lifetime as well.

1. Introduction

Every processor chip has a physical limit on power dissipation it can support. For systems that use these processors, performance and power become opposing requirements. Modern computing systems, therefore, have built-in power control schemes. For example, thermal sensors on a processor chip may trigger a slowdown of the processor clock [30].

For mobile systems, energy consumption and the rate of consumption (power) are directly related to the battery capacity. Higher discharge rate reduces the capacity, requiring bulkier batteries with higher current rating [4] or more frequent recharging. Thus, it is important to control the power consumption. On the other hand, during the idle periods most systems may be put in energy-saving standby mode. When excessive power consumption forces CSD, the completion time of the ongoing system task increases. This increases the energy consumption. The energy penalty of the CSD method can be severe for high-leakage technologies. CSD is, therefore, not recommended without voltage scaling [10]. There is, however, another consideration also. The reduced power slows the current drain from the battery. For a given battery capacity, this can increase the lifetime of the battery [17, 25]. *Lifetime* here refers to the useful life of a *primary* battery or the time between recharges for a *secondary* (rechargeable) battery. If the increase in the battery lifetime for a portable device is more than the increase in the execution time of the task, then CSD can be beneficial [3]. Unless the efficiency aspect of the power source is properly considered, the

slowing down of a computing task for power reduction would not be recommended. The lack of such consideration often results in the use of oversize batteries as well as excess design for unnecessary power dissipation, cooling, etc.

In the next section, we discuss a scenario where CSD may be necessary. We also find that its power saving advantage diminishes in higher leakage technologies. This leads to our motivation for finding a lower energy penalty alternative. Because CSD allows larger delay for hardware, we can further reduce power by lowering the supply voltage. Voltage reduction reduces both power and energy. However, this has limited potential in the nanometer technologies where the voltage, already lowered due to the electric field requirement, is closer to the threshold voltage. This is particularly so for dual-threshold designs in which high-threshold devices are used to reduce leakage.

When voltage has been scaled down to some limit set by the technology, further power reduction, if necessary due to the system or operational requirements, by CSD will increase the task completion time and the leakage energy. To reduce the energy, a *dynamic power cutoff technique* (DPCT) has been proposed [32]. While DPCT can save both power and energy, it requires turning power off and on for different parts of combinational logic at different times within the clock period. Asynchronous delays for power control signals make the design complex and especially sensitive to process variation.

In this paper, we address the need for a power saving method with emphasis on the energy penalty. We propose an *instruction slowdown* (ISD) method, which inserts NOP-like instructions. A new instruction named SLOP (slowdown for low power) is automatically inserted by the processor control that also generates power-down, sleep mode, or clock gating signals for various hardware units. We have analyzed several technologies ranging through to 32nm and shown that the ISD method is equally or more effective than the CSD method.

In general, the slowdown of a computing task can consume more energy. In fact, it would always turn out to be that way if we considered the raw energy consumption from an ideal source. The conclusions differ when we consider a real source, such as the battery in a portable device. A relevant parameter is the *lifetime* or the time between consecutive recharging of a battery. A battery's capacity, usually in mA-hours, is a valid indicator of the recharge time if the battery supplies close to the rated current. At higher currents, the capacity degrades. Thus, reduction in power consumption (or current drain) can enhance the lifetime [25]. We use a battery model based on the classical Peukert's law [17] to represent the battery lifetime, which is adjusted for the increased task execution time. Alternatively, a battery efficiency model [22] can also be used. Slowdown for power

reduction is considered beneficial only if the adjusted lifetime is enhanced. This advantage of ISD becomes more pronounced as the technology becomes leakier.

ISD can be compared to another proposed power saving method called *fetch throttling* [28, 29]. This method, when applied to multiple issue processors, slows down the rate of instruction fetch based on the lack of any parallel execution opportunity in the program being executed. Thus, the instructions that would have waited in the pipeline due to data, resource, or control conflicts are fetched after suitable delays. The reported average reduction in energy delay product is 6.7% for static throttling and could go up to 15% with dynamic throttling. These savings are due to the avoidance of incorrect speculations. We can reduce the performance penalty of ISD by inserting the NOPs after those instructions that require speculation. However, this aspect is not discussed in this paper and should be explored in the future. The objective of the present work is to reduce power with minimal energy cost.

In this paper, Section 2 gives the problem statement and background. We have analyzed the CSD method and cited prior work on the NOP instruction for power. Section 3 gives an analysis of the new ISD method and proposes the SLOP instruction. Section 4 discusses hardware implementation of SLOP and estimates the relevant parameters, leakage factor k and SLOP power factor β , for several CMOS technologies. Section 5 presents computed results on power saving and battery lifetime for various technologies. Section 6 proposes unexplored possibilities.

2 Background

Consider a processor built in certain semiconductor technology. If we reduce the supply voltage V , the critical path delay will increase and hence the maximum clock frequency f will have to be decreased. This will reduce the dynamic power in proportion to V^2f . Static power will also decrease as V^2 . However, a measure of energy a computing task will use is the total energy per cycle (EPC), consisting of dynamic EPC and static EPC. Dynamic EPC is proportional to V^2 and static EPC is proportional to V^2/f . We notice that dynamic EPC always reduces with voltage scale down. However, static EPC is proportional to $1/f$, which will increase rapidly as V approaches close to the threshold voltage.

Thus, for a given technology (i.e., given threshold voltage), there is an optimum supply voltage and a corresponding clock frequency that minimize the total EPC. Any further power reduction by voltage scaling beyond this optimum value will incur an increase in the total EPC, although power will reduce. As the supply voltage gets closer to the threshold voltage, the performance also becomes sensitive to process variation that is common in nano-scale technologies. In practice, therefore, the supply voltage has a lower bound. If further power reduction is required, say, due to battery characteristics, thermal factors or other operational considerations, then clock frequency alone would have to be reduced. This will reduce power but increase EPC. Dynamic voltage control within a clock period [32] can reduce the EPC but as pointed out earlier, requires complex control circuitry.

We assume a situation where voltage is at its lowest permissible limit and power must be reduced. Traditionally,

we would slowdown the clock and let EPC increase. This will be a performance-power trade-off that involves an essential energy penalty. We explore an alternative solution in which clock is not slowed down but performance is traded off, similar to clock slowdown, for power reduction while energy penalty is reduced, especially for high leakage technologies.

CSD is a known technique for power reduction and we use it as a reference for evaluating the proposed method. When we slow down the clock, dynamic power is reduced in proportion to the clock rate whereas leakage power remains unchanged. The computing task now takes longer to complete. This results in the same dynamic energy consumption whereas the leakage energy consumed is more. We will use a processor slowdown factor n . Without loss of generality, n is assumed to be an integer. Thus, $n = 1$ is the normal (rated-clock) operation. Let us define:

$$n = \text{processor slowdown factor} \quad (1)$$

$$f = \text{rated clock frequency in Hz} \quad (2)$$

$$P_d = \text{dynamic power with rated clock} \quad (3)$$

$$P_s = \text{static power with rated clock} \quad (4)$$

$$k = P_s/P_d = \text{static power ratio} \quad (5)$$

$$T = \text{time duration of a computing task} \quad (6)$$

When the processor is slowed down n times, its power consumption is given by,

$$P_{CSD}(n) = \frac{P_d}{n} + P_s = P_d \frac{1 + kn}{n} \quad (7)$$

We notice that a computing task of original duration T is now completed in duration nT . However, we may expect that a reduced current from the battery will result in an enhanced capacity to supply energy and increase the lifetime, L . This is often represented by Peukert's law [17, 25]:

$$L = C_1/I^\alpha = C_2/P^\alpha \quad (8)$$

where C_1 and C_2 are constants related to the battery capacity, I is the current, and P is power assumed to be drawn at a constant rated voltage. In reality, this formula assumes a constant current. Though not a reality for digital circuits, this condition can be maintained by using a *supercapacitor* and battery combination [26]. In this case, the current fluctuations are smoothed by a large capacitor of several farads capacity. The exponent in equation 8 can take different values depending on the type of battery, for the present illustration we use $\alpha = 1.3$.

Next, we denote the power and battery lifetime savings by the following ratios:

$$P_{CSDratio} = \frac{P_{CSD}(n)}{P_{CSD}(1)} = \frac{1 + kn}{n(1 + k)} \quad (9)$$

$$L_{CSDratio} = \frac{1}{n} \times \frac{1}{(P_{CSDratio})^\alpha} \quad (10)$$

From these equations, we get,

$$E_{CSDratio} = nP_{CSDratio} \quad (11)$$

We observe that for very low leakage, $k \approx 0$, $P_{CSDratio} = 1/n$ and $L_{CSDratio} = n^{0.3}/(1 + n)$,

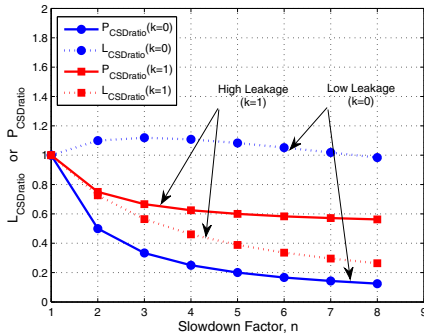


Figure 1. CSD power and battery lifetime ratios.

which show power saving with lifetime enhancement at least for small values of n . To consider very high leakage technologies, let us assume $k = 1$. Then $P_{CSDratio} = (1 + n)/(2n)$. CSD now cannot reduce the power ratio below 0.5 and there is battery lifetime degradation for any clock slowdown factor n . These trends are illustrated in Figure 1.

In the next section, we will introduce a new power reduction method called ISD. The processor is slowed down not by clock slowdown but by inserting NOP cycles. The NOP instruction has been used for power optimization. Najeb *et al.* [20] mix NOP instructions in an instruction sequence to produce a maximum power consuming cycle, which they term as *power virus*. Such an instruction sequence is useful for the design and test of the processor. Lotfi-Kamran *et al.* [18] suggest freezing certain data bits in a pipeline processor whenever a NOP, either contained in the instruction stream or generated due to hazards, is executed. They report about 10% power saving with a modest hardware overhead of 0.1%. Hurd [13] describes a technique of manipulating the positions of NOP instructions in a multiple instruction word architecture so that certain instructions need not be fetched. In another technique, also due to Hurd [12], a NOP instruction is replaced by another instruction called “proxy NOP”. This instruction uses the data patterns of its neighboring instruction but executes like NOP. It thus reduces activity in the datapath. None of these techniques perform the power management as discussed in the following section.

3 Instruction Slowdown (ISD)

In this new methodology, the operation of a processor is slowed down for power reduction by inserting non-functional cycles while the rated clock frequency (f) is maintained. This is similar to inserting instruction we call SLOP (slowdown for low power). Although it is described as a purely hardware induced operation, SLOP can be included in the software instruction set.

In a typical implementation, a power management unit (PMU) monitors the system and, if necessary, determines an appropriate slowdown factor (n), which is supplied to the control. The control then inserts the required number of SLOPs in the pipeline. The factor n is assumed to be an integer here but, in general, can be any number that determines the percentage of SLOPs to inserted in the instruction stream.

Hardware execution of SLOP resembles a conventional NOP, stall or bubble [21] with a few differences. First, its execution in a pipeline requires no “fetch”

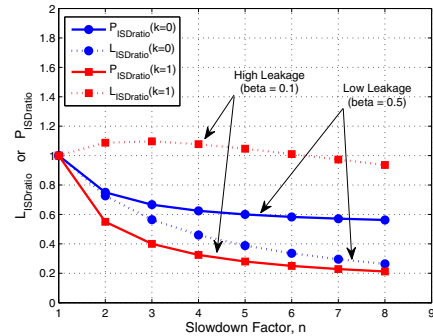


Figure 2. ISD power and battery lifetime ratios.

because the control generates it locally. Second, the control generates low power mode signals for various hardware units. To analyze the power and energy relations, we will use the same symbol definitions as in the previous section. We also define a SLOP power factor:

$$\beta = \frac{\text{power consumed by SLOP}}{\text{av. power consumed by non NOP instr.}} \quad (12)$$

where $0 \leq \beta \leq 1$. For a slowdown factor n , we insert $n-1$ SLOPs after each instruction. Consider a period of 1 second, containing f clock cycles. The energy consumed during a regular instruction (assumed to be non-NOP) cycle is $P_d(1+k)/f$ and that during a SLOP cycle is $\beta P_d(1+k)/f$. Of those f cycles, f/n are regular instruction cycles and $(n-1)f/n$ are SLOP cycles. Thus, total power consumption, or energy dissipated per second, is obtained as,

$$\begin{aligned} P_{ISD}(n) &= \frac{P_d(1+k) \frac{f}{n} + \beta P_d(1+k) \frac{(n-1)f}{n}}{f} \\ &= P_d(1+k) \frac{\beta n - \beta + 1}{n} \end{aligned} \quad (13)$$

Similar to the CSD, now also a computing task of original duration T will require nT time. We find the power and battery lifetime ratios as follows:

$$P_{ISDratio} = \frac{P_{ISD}(n)}{P_{ISD}(1)} = \frac{\beta n - \beta + 1}{n} \quad (14)$$

$$L_{ISDratio} = \frac{1}{n} \times \frac{1}{(P_{ISDratio})^\alpha} \quad (15)$$

These lifetime and power ratios as functions of slowdown factor n are shown in Figure 2. A ratio below 1 indicate both power reduction (desirable) and lifetime reduction (undesirable). Notice that power (solid line) is always reduced. More reduction is achieved for higher leakage ($\beta = 0.1$) technology. Lifetime (dotted line) for high leakage improves for small n and then degrades because the NOP cycles consume non-zero energy. However, the lifetime degrades for low leakage technology in a similar way as it did for CSD with high leakage.

4 SLOP in Hardware

We used a 32-bit MIPS pipelined processor for evaluation of the ISD and CSD methods. It has a

Table 1. HSPICE simulation (32nm CMOS, 90°C).

Hardware block	Energy/cycle		SLOP power		
	Dyn. nJ	Stat. nJ	Power mode	Dyn. %	Stat. %
PC	85114	17742	CG	25	100
PC+1 adder	28947	6536	PG	0	0
IM	6780	3209	Drowsy	25	25
Regfile	98262	192375	CG	30	100
Forwarding	31297	4090	PG	0	0
Hazard	25421	3744	PG	0	0
Controller	14338	2973	None	100	100
32-b ALU	263815	22346	PG	0	0
32-b comp	39710	5695	PG	0	0
DM	64343	50699	Drowsy	25	25
3-1 mux	392374	56299	PG	0	0
2-1 mux	204456	44106	PG	0	0
BrnchAddrCal	181878	13680	PG	0	0
IF/ID reg	156027	32048	CG	50	100
ID/EX reg	213447	58412	CG	50	100
EX/DM reg	131023	34324	CG	50	100
DM/WB reg	127885	33481	CG	50	100
ForwDM/WB	5820	1009	PG	0	0

```

0000 LW $1, X:0002($0)
0001 ADD $4, $1, $0
0002 ADD $1, $0, $0
0003 LW $3, X:0004($0)
0004 LW $2, X:0003($0)
0005 BEQ $2, $0, X:0003
0006 SUB $2, $2, $3
0007 ADD $1, $1, $4
0008 J X:0000005
0009 SW $1, X:0004($3)
000A #J X:000000A(HALT)

```

Figure 3. MIPS program used power estimation.

conventional five-stage pipeline containing the fetch (IF), decode (ID), execute (EX), memory (DM) and write-back (WB) stages [21]. It also contains hazard and forwarding units. We obtained an available VHDL model [2] and synthesized using Mentor Graphics Leonardo Spectrum. This provided us a gate-level model for power analysis.

Various blocks of the processor were extracted as transistor-level netlists using Mentor Graphics Design Architect. Each block was simulated in HSPICE for 1,000 random input vectors with 10ns clock rate ($f = 100\text{MHz}$) to determine the average per cycle dynamic and static energy dissipation. This evaluation was repeated for five CMOS technologies, 180nm, 90nm, 65nm, 45nm and 32 nm, using the predictive technology models (PTM) [1, 5, 33]. The simulation assumed 90°C temperature. A sample result for 32nm is shown in Table 1. The last three columns of this table are discussed later. Communication buses are not considered separately because all drivers and buffers are included as parts of various hardware blocks.

We wrote a MIPS program that multiplies hexadecimal integers FFFF and 0004 by repeated additions. Our processor has separately addressable instruction (IM) and data (DM) memories. Initially, $\text{DM}(2) = \text{FFFF}$, $\text{DM}(3) = 4$, $\text{DM}(4) = 1$. Final result is $\text{DM}(5) = 0003\text{FFFC}$. The MIPS code is given in Figure 3.

This program completes in 34 cycles. The number of times pipeline stages are activated are: 34 IF, 29 ID, 18 EX, 4 DM and 14 WB. The execution statistics of hardware stages and the instruction mix as well as the number of cycles can be easily changed by varying the parameters in the program. It was assembled by hand and the gate-level model was simulated using Mentor Graphics ModelSim. The final result was verified. For power, active blocks in a pipeline stage were identified. Total energy of the pipeline stage was computed by adding the dynamic and static energies of its active blocks. After characterizing each pipeline

Table 2. Leakage factor (k) and SLOP power factor (β).

Technology	Leakage factor k	SLOP power factor β
180nm	0.097	0.265081
90nm	0.124	0.23699
65nm	0.268	0.212003
45nm	0.353	0.183881
32nm	0.413	0.159012

stage for its energy, the total energy of the program was computed by adding energies of pipeline stages as per the numbers obtained above. The dynamic energy was added up for active stages while the static energy was added up for all blocks for 34 cycles, using the technology-specific data (e.g., Table 1 for 32nm). The ratio of total static energy to dynamic energy for each technology gives the respective value of the leakage factor k shown in Table 2.

Table 1 quantitatively shows how power was reduced by clock gating (CG), power gating (PG) and drowsy memories. Power gating (PG) focuses on leakage. Circuit level approaches for leakage reduction include body bias control [7], dual threshold domino logic [6, 15], input vector control [14] and power gating [11, 16, 24]. We adopt power gating for combinational blocks. It is assumed that the supply line will be gated by pull-up or a pull-down devices that will be put in the cutoff mode during SLOP cycles. This will almost completely eliminate both static and dynamic power during those cycles [9]. We must, however, realize that power gating at clock cycle level represents a design challenge. Studies [7, 27] show that improvements will be needed both in the speed and energy cost of power control and implemented in the present-day design.

Drowsy mode for caches: The dynamic and leakage power consumed by instruction and data caches is a sizable portion of total power consumed by the processor. During SLOP cycles, the memory cells are put into low voltage “drowsy mode”, which can allow up to 75% of energy reduction with no more than 1% of performance overhead [8]. In addition, decoder and sense amplifier can be power gated. Another technique identifies an application’s cache requirements dynamically, and uses a circuit-level mechanism, “*gated-Vdd*”, to gate the supply voltage to the SRAM cells of the cache’s unused sections to reduce leakage [24].

Clock gating (CG) is applied to registers. Their power is not gated because the state must be preserved. A significant fraction of the dynamic power in a processor is consumed by the clock network and flip-flops. The clock buffers can consume 50% or more of total dynamic power [16, 31]. Clock gating turns off the clocks when they are not required or stop them from feeding to the components which are not being used. Results show that up to 43% power saving can be achieved with a possible 20% reduction in area when clock gating replaces the state-retention feedback logic of flip-flops [23]. The clock gating employed in the register file with high switching activity of about 0.25 shows that power saving of about 70% can be achieved [19].

At the time of this writing, we have not completed an evaluation of these techniques. The data in the last two columns of Table 1 is based on the references cited here. To compute the SLOP power factor (β) we first weight columns 2 and 3 by columns 5 and 6, respectively. The dynamic and static power of a

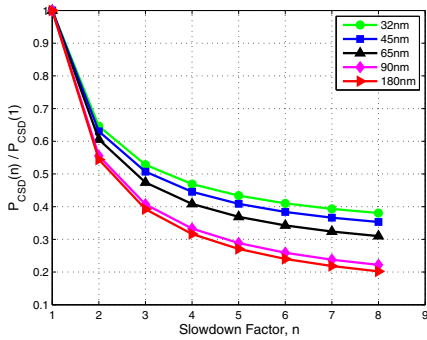


Figure 4. CSD power ratios.

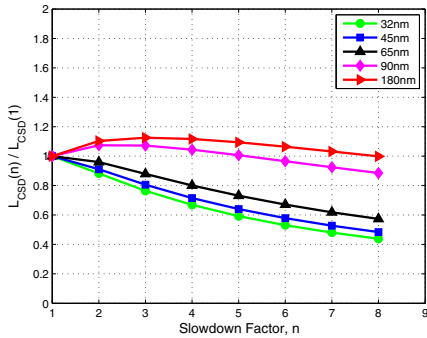


Figure 5. CSD battery lifetime ratios.

SLOP cycle is then calculated in a similar way as described before for a regular instruction. The ratio of the power of SLOP cycle to that of the regular instruction cycle is β given in Table 2.

5 Results

Figures 4 and 5 display power and battery lifetime ratios as functions of the CSD factor n for five CMOS technologies. These graphs were computed from equations 9 and 10, respectively, using values of leakage factor k taken from Table 2. We observe that the CSD method degrades for technologies that are finer than 65nm. This is because as n increases, leakage power becomes a dominant factor in the total power. Besides, saving of dynamic energy is compensated for by increase of leakage energy.

Figures 6 and 7 display power and battery lifetime ratios as functions of the ISD factor n for five CMOS technologies. These graphs were computed from equations 14 and 15, respectively, using values of SLOP power factor β taken from Table 2. Because ISD is assisted by hardware in reducing leakage for the SLOP cycles, we see greater savings of power for high leakage 32nm technology.

To compare the two methods directly, we use equations 7 and 8 to obtain the following ratio:

$$\frac{P_{CSD}}{P_{ISD}} = \frac{1 + kn}{(1 + k)(\beta n - \beta + 1)} \quad (16)$$

The graph in Figure 8 shows this ratio as a function of the slowdown factor n for five technologies in the range 180nm through 32nm. The ratio = 1 horizontal line divides this graph in two parts. Points above this line favor ISD and those below favor CSD. The curves will shift upward with improved dynamic power management in high leakage technologies. Results for battery lifetime are shown in Figure 9.

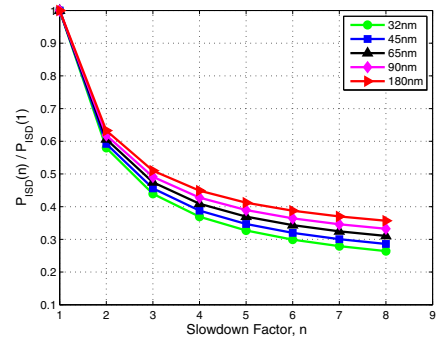


Figure 6. ISD power ratios.

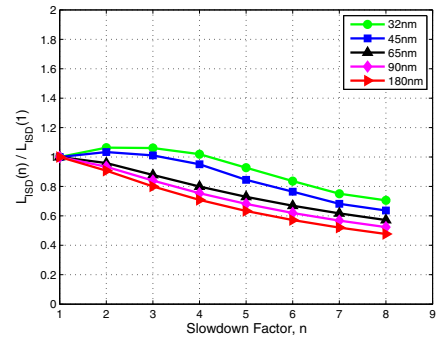


Figure 7. ISD energy ratios.

6 Conclusion

The proposed ISD has advantages in power saving for high leakage technologies. We suggest combining the slowdown methods with overall supply voltage scaling. Voltage reduction will save dynamic and static power as well as energy. But the increased hardware delay will necessitate a clock slowdown. Thus, for $n = 2$ CSD may be used. Thereafter, $n > 2$ slowdown should use ISD. The throughput aspect of slowdown methods is not studied. CSD preserves all hazard penalties and throughput drops as $1/n$. ISD will eliminate hazards progressively as n increases. SLOP is presented purely as an internal mechanism supported by power management and control hardware. Its inclusion in the instruction set will allow compilers to explore creative ways to use the power management hardware.

References

- [1] <http://www.eas.asu.edu/~ptm>.
- [2] A. Arthurs and L. Ngo, "Analysis of the MIPS 32-Bit, Pipelined Processor Using Synthesized VHDL," Technical report, University of Arkansas, Department of Computer Science and Engineering, www.csce.uark.edu/~ajarthu/papers/mips-vhdl.pdf.
- [3] L. Benini and G. D. Micheli, *Dynamic Power Management, Design Techniques and CAD Tools*. Springer, 1998.
- [4] I. Buchmann, *Batteries in a Portable World: A Handbook on Rechargeable Batteries for Non-Engineers*. Richmond, British Columbia: Cedex Electronics, Inc., second edition, 2001.
- [5] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New Paradigm of Predictive MOSFET and Interconnect Modeling for Early Circuit Design," in *Proc. Custom Integrated Circuits Conf.*, 2000, pp. 201-204.

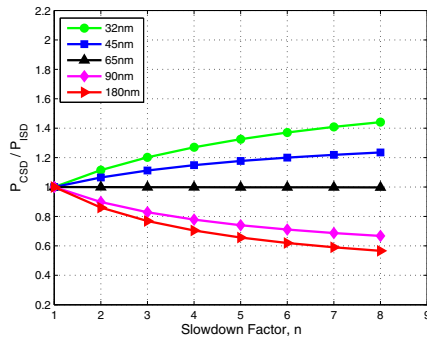


Figure 8. CSD vs. ISD power ratios.

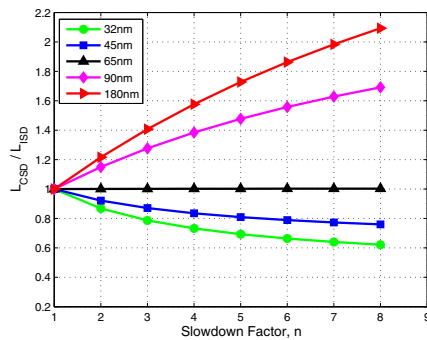


Figure 9. CSD vs. ISD battery lifetime ratios.

- [6] S. Dropsho, V. Kursun, D. H. Albonese, S. Dwarkadas, and E. G. Friedman, "Managing Static Leakage Energy in Microprocessor Functional Units," in *Proc. 35th Annual International Symp. Microarchitecture, MICRO*, 2002, pp. 321–332.
- [7] D. Duarte, Y. F. Tsai, N. Vijaykrishnan, and M. J. Irwin, "Evaluating Run-Time Techniques for Leakage Power Reduction," in *Proc. 15th International Conf. VLSI Design*, 2002.
- [8] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge, "Drowsy Caches: Simple Techniques for Reducing Leakage Power," in *Proc. International Symposium on Computer Architecture*, 2002, pp. 148–157.
- [9] J. Frenkil and S. Venkatraman, "Power Gating Design Automation," in D. Chinnery and K. Keutzer, editors, *Closing the Power Gap Between ASIC & Custom Tools and Techniques for Low-Power Design*, chapter 10, pp. 251–280, Springer, 2007.
- [10] M. Horowitz, T. Indermaur, and R. Gonzalez, "Low-Power Digital Design," in *Proc. International Symp. Low Power Electronics and Design*, 1994, pp. 8–11.
- [11] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural Techniques for Power Gating of Execution Units," in *Proc. International Symp. Low Power Electronics and Design*, 2004, pp. 32–37.
- [12] L. L. Hurd, "Power Reduction for Multiple-Instruction-Word Processors with Proxy NOP Instructions." U.S. Patent 6535984. March 18, 2003.
- [13] L. L. Hurd, "Power Saving by Disabling Memory Block Access for Aligned NOP Slots During Fetch of Multiple Instruction Words." U.S. Patent 6442701. August 27, 2002.
- [14] M. C. Johnson, D. Somasekhar, L.-Y. Chiou, and K. Roy, "Leakage Control with Efficient Use of Transistor Stacks in Single Threshold CMOS," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 10, no. 1, pp. 1–5, Feb. 2002.
- [15] J. T. Kao and A. P. Chandrakasan, "Dual-Threshold Voltage Techniques for Low-Power Digital Circuits," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 7, pp. 1009–1018, July 2000.
- [16] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, *Low Power Methodology Manual for System On Chip Design*. Boston: Springer, 2008.
- [17] D. Linden and T. Reddy, *Handbook of Batteries, 3rd Edition*. McGraw-Hill, 2001.
- [18] P. Lotfi-Kamran, A. Rahmani, A. Salehpour, A. Afzali-Kusha, and Z. Navabi, "Stall Power Reduction in Pipelined Architecture Processors," in *Proc. of 21st International Conference on VLSI Design*, 2008, pp. 541–546.
- [19] M. Mueller, A. Wortmann, S. Simon, M. Kugel, and T. Schoenauer, "The Impact of Clock Gating Schemes on the Power Dissipation of Synthesizable Register Files," in *Proc. International Symp. Circuits and Systems*, volume 2, 2004, pp. 609–612.
- [20] K. Najeeb, V. V. R. Konda, S. S. Hari, V. Kamakoti, and V. M. Vedula, "Power Virus Generation Using Behavioral Models of Circuits," in *Proc. 25th IEEE VLSI Test Symposium*, 2007, pp. 35–40.
- [21] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface, Fourth Edition*. Morgan Kaufmann, 2009.
- [22] M. Pedram and Q. Wu, "Design Considerations for Battery-Powered Electronics," in *Proceedings 36th Design Automation Conference*, June 1999, pp. 861–866.
- [23] K. C. Pokhrel, "Physical and Silicon Measures of Low Power Clock Gating Success: An Apple to Apple Case Study." Synopsys Users Group (SNUG), 2007.
- [24] M. Powell, S.-H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar, "Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories," in *Proc. International Symp. Low Power Electronics and Design*, 2000, pp. 90–95.
- [25] R. Rao, S. Vrudhula, and D. N. Rakhmatov, "Battery Modeling for Energy-Aware System Design," *Computer*, vol. 36, no. 12, pp. 77–87, Dec. 2003.
- [26] R. F. Service, "New 'Supercapacitor' Promises to Pack More Electrical Punch," *Science*, vol. 313, p. 902, 18 Aug. 2006.
- [27] J. W. Tschanz, S. G. Narendra, Y. Ye, B. A. Bloechel, S. Borkar, and V. De, "Dynamic Sleep Transistor and Body Bias for Active Leakage Power Control of Microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 11, pp. 1838–1845, Nov. 2003.
- [28] O. S. Unsal, I. Koren, C. M. Krishna, and C. A. Moritz, "Cool-Fetch: Compiler-Enabled Power-Aware Fetch Throttling," *IEEE Computer Architecture Letters*, vol. 1, Apr. 2002.
- [29] H. Wang, Y. Guo, I. Koren, and C. M. Krishna, "Compiler-Based Adaptive Fetch Throttling for Energy-Efficiency," in *IEEE International Symp. on Performance Analysis of Systems and Software*, Mar. 2006, pp. 112–119.
- [30] W. Wolf, "Cyber-Physical Systems," *Computer*, vol. 42, no. 3, pp. 88–89, Mar. 2009.
- [31] K.-S. Yeo and K. Roy, *Low-Voltage, Low-Power VLSI Subsystems*. McGraw-Hill, 2005.
- [32] B. Yu and M. L. Bushnell, "A Novel Dynamic Power Cutoff Technique (DPCT) for Active Leakage Reduction in Deep Submicron CMOS Circuits," in *Proc. International Symp. Low Power Electronics and Design*, 2006, pp. 214–219.
- [33] W. Zhao and Y. Cao, "New Generation of Predictive Technology Model for Sub-45nm Early Design Exploration," *IEEE Transactions on Electron Devices*, vol. 53, pp. 2816–2823, Nov. 2006.